Q1. In Python 3.X, what are the names and functions of string object types?

* str: This is the most basic string type. It is a sequence of characters that can be used to represent text.
* bytes: This type is used to represent binary data. It is a sequence of bytes that can be used to represent images, sounds, and other types of data.
* unicode: This type is used to represent text in Unicode format. Unicode is a standard that allows for the representation of text in a wide variety of languages.
* strp: This type is used to represent text that has been stripped of whitespace. Whitespace is any character that is considered to be a space, including spaces, tabs, and newlines.
* strip: This type is used to represent text that has been stripped of whitespace from both ends.
* replace: This type is used to replace a substring in a string with another substring.
* split: This type is used to split a string into a list of strings based on a delimiter. A delimiter is a character that is used to separate the strings in the list.
* join: This type is used to join a list of strings into a single string.
* find: This type is used to find the first occurrence of a substring in a string.
* rfind: This type is used to find the last occurrence of a substring in a string.
* index: This type is used to find the index of the first occurrence of a substring in a string.
* rindex: This type is used to find the index of the last occurrence of a substring in a string.
* count: This type is used to count the number of occurrences of a substring in a string.
* startswith: This type is used to check if a string starts with a specified substring.
* endswith: This type is used to check if a string ends with a specified substring.
* isalnum: This type is used to check if all the characters in a string are alphanumeric.
* isalpha: This type is used to check if all the characters in a string are alphabetic.
* isdigit: This type is used to check if all the characters in a string are digits.
* islower: This type is used to check if all the characters in a string are lowercase.
* isupper: This type is used to check if all the characters in a string are uppercase.
* istitle: This type is used to check if all the characters in a string are titlecase.
* isprintable: This type is used to check if all the characters in a string are printable.
* isspace: This type is used to check if all the characters in a string are whitespace.

Q2. How do the string forms in Python 3.X vary in terms of operations?

* Mutable strings:

In Python 2.X, strings were mutable. This meant that you could change the contents of a string after it had been created. This was often used for things like concatenating strings together or replacing parts of a string.

* Immutable strings:

In Python 3.X, strings are immutable. This means that you cannot change the contents of a string after it has been created. This was done to make Python more consistent and to prevent errors from occurring when trying to change a string that is already in use.

* Unicode strings:

In Python 3.X, strings are Unicode strings. This means that they can contain characters from any language in the world. This was done to make Python more internationalized and to allow it to be used with languages that have non-ASCII characters.

* Byte strings:

In Python 3.X, there is also a type of string called a byte string. This is a string that contains binary data. This is often used for things like storing images or audio files.

Q3. In 3.X, how do you put non-ASCII Unicode characters in a string?

The syntax is very simple. We need to add one escape sequence for the character we want to print. Let's see the implementation. In this code, the one '\u03A9' is an escape sequence that is used to represent the 'omega' character in the Non-ASCII character sequence.

Q4. In Python 3.X, what are the key differences between text-mode and binary-mode files?

* Text mode: is the default mode for opening files in Python. It is used for reading and writing text files, such as strings or characters. In text mode, Python automatically handles the encoding and decoding of the data, depending on the platform's default encoding scheme.
* Binary mode: is used for reading and writing binary files, such as images or audio files. In binary mode, Python does not perform any encoding or decoding of the data. The data is written to the file exactly as it is received from the input stream.

Q5. How can you interpret a Unicode text file containing text encoded in a different encoding than your platform's default?

One way is to use the open() function in Python with the encoding parameter set to the encoding of the file. For example, to open a file encoded in UTF-8, you would use the following code:

open("file.txt", "r", encoding="utf-8")

Q6. What is the best way to make a Unicode text file in a particular encoding format?

To write a file in Unicode (UTF-8) encoding in Python, you can use the built-in open() function with the 'w' mode and specifying the encoding as "utf-8".

Here's an example: with open("file. txt", "w", encoding="utf-8") as f: f.

Q7. What qualifies ASCII text as a form of Unicode text?

ASCII also uses 7 and 8 bits for the representation of characters. A large number of characters used around the world which cannot be encoded by using 8-bit representation led to the creation of UTF-16 and UTF-32 encoding formats under Unicode encoding. Thus, ASCII is a subset of the Unicode encoding scheme.

Q8. How much of an effect does the change in string types in Python 3.X have on your code?

The change in string types in Python 3.X has a significant effect on your code if you are using Python 2.7 or earlier. In Python 3, strings are now Unicode objects, which means that they can contain characters from any language. This is a major change from Python 2, where strings were ASCII objects, which could only contain characters from the ASCII character set.